Neo N3 White Paper

Version 1.0
June 28, 2022

Table of Contents

Preface

A blockchain is a decentralized distributed ledger system. It can be used for registration and issuance of digitized assets, property right certificates, credit points, and so on. Blockchains enable transfers, payments, and transactions in a peer-to-peer manner. Blockchain technology was first proposed by Satoshi Nakamoto in a cryptography mailing list, i.e. Bitcoin. Since then, numerous applications based on blockchain have emerged, such as e-cash systems, stock equity exchanges, and Smart Contract systems.

A blockchain system is advantageous over a traditional centralized ledger system for its full-openness, immutability, resistance to double-spending, and a lack of reliance on any kind of trusted third-party.

Like all distributed systems, blockchain systems are challenged with network latency, transmission errors, software bugs, security loopholes and black-hat hacker threats. Moreover, its decentralized nature suggests that no participant of the system can be trusted. Malicious nodes may emerge, as well as data differences due to conflicting interests.

To counter these potential errors, a blockchain system is in need of an efficient consensus mechanism to ensure that every node on the network has a copy of the ledger which is globally recognized as the truthful version. Traditional fault tolerance mechanisms concerning certain problems may not be completely capable of tackling the issues that distributed blockchain systems are faced with. A universal cure-to-all fault tolerance solution is needed.

Nakamoto Consensus and the Proof-of-Work mechanism, employed by Bitcoin, addresses this issue rather brilliantly. But it comes with an obvious price, i.e. significant electricity costs and energy consumption. Further, with Bitcoin's existence, new blockchains must find different hashing algorithms so as to prevent computational attacks from it. For example, Litecoin adopts SCRYPT, rather than Bitcoin's SHA256.

Neo's Vision

Neo (formerly Antshares) is an open-source decentralized blockchain application platform founded in 2014 by Da HongFei and Erik Zhang. Since its rebranding to Neo from

Antshares in 2017, the project's vision is to realize a "smart economy" by utilizing blockchain technology and smart contracts to issue and manage digitized assets.

The Neo network runs on the Proof-of-Stake-based delegated Byzantine fault tolerant (dBFT) consensus mechanism between a number of globally elected nodes. The base asset of the Neo blockchain is the non-divisible NEO token, which is used for governance and generates GAS tokens. These GAS tokens, a separate asset on the network, can be used to pay for transaction fees, and are divisible down to the smallest unit 0.00000001. The inflation rate of GAS is controlled by a council formed from the top 21 elected nodes.

A total of 100 million Neo were created in the Genesis Block. 50 million Neo were sold to early investors through an initial coin offering in 2016 that raised US 4.65 million, while the remaining 50 million Neo were locked for gradual distribution. Each year, a maximum of 15 million Neo tokens are unlocked for use by the Neo development team to fund the platform’s long-term development and ecosystem growth.

The core of the Neo feature set revolves around tools that allow developers to efficiently deploy and scale smart contract applications on the Neo blockchain. Developers are able to build their applications in a wide range of popular languages and take advantage of the powerful native services.

History of Neo

From the beginning, Neo’s vision has been to build the most developer-friendly blockchain. This vision started with Antshares and Neo Legacy. In 2014, Antshares was founded by Da Hongfei and Erik Zhang. In the following year, it was open-sourced on GitHub and by September 2015, the Neo Legacy white paper was released. Neo was officially rebranded from Antshares in June 2017, with the idea of combining the past and the future. The word “neo” originates from the Greek word “νέο“, meaning “new”, “modern”, and “young”. The vision of building a “smart economy” was developed alongside the rebranding.

Enter Neo N3. Neo 3.0, or N3, was first announced by Erik Zhang in July, 2018 as an upgrade to the Neo Legacy protocol. N3 improves on previous versions; it has more powerful, robust, and complete features, plus a highly modular architecture.

Due to the extent of improvements implemented in the N3 update, certain features are not backwards compatible with the Neo Legacy blockchain. Therefore the N3 updates were implemented as a new blockchain to which Legacy users could migrate their assets.

N3 delivers numerous features, including off-chain decentralized storage and platform native data oracles. It brings many fundamental upgrades from Neo Legacy, including a revamped governance system, an improved economic model, and an all-new account model to overcome the limitations of the previous UTXO-based system. Its multi-language support and best-in-class tooling make Neo a compelling choice for blockchain developers and traditional software engineers alike.

Technical Overview

Introduction

A blockchain is a distributed ledger system in which participants connect with each other in a peer-to-peer network. All messages within it will be sent by broadcasting. Two types of roles exist: ordinary nodes and consensus nodes. Ordinary nodes use the system to transfer and exchange, accepting ledger data; while consensus nodes are tasked with maintaining the ledger by approving new blocks, ensuring correctness.

Hypothetically, in this system, messages may be subject to loss, damage, latency and repetition. The sending order may not necessarily be consistent with the receiving order of messages. The activities of nodes could be arbitrary; they may join and quit the network at any time. They may also dump and falsify information or simply stop working. Artificial or non-artificial glitches may also occur.

The integrity and authenticity of information transmission are ensured with cryptography while senders must attach signatures to the hash value of the message sent.

Blockchain Models

A blockchain is a data structure. Blocks are composed of a block header and a block body. As each block contains a cryptographic hash of the previous block, a chain structure is formed. This prevents arbitrary modification of past blocks, as changes to previous blocks would invalidate the hashes of future blocks.

The block header contains the basic information of a block and provides verification of a block, allowing for its verification. The block body is a transaction list, which essentially starts with the transaction list length, followed by a list of transactions. At present, there can be up to 512 transactions per block, however this parameter may be adjusted depending on network demand to allow greater throughput.

Neo Token Models

Native Tokens

There are two kinds of native tokens defined in the Neo system: NEO and GAS.

NEO is the governance token. NEO holders can participate in Neo network management by voting for candidate nodes. The top 21 voted candidate nodes form the Neo Council, who are responsible for network parameter modification and participate in consensus. The total amount of NEO is 100 million. Its minimum unit is 1 and can not be divided. This supply is registered in the Genesis block and initially stored within a multi-signature address controlled by the standby validators.

GAS is the fuel token of the network, with a smallest unit of 0.00000001. Users can obtain GAS either through a claim or purchase. When using the Neo network, they need to pay a certain amount of GAS in the form of system and network fees. All transactions require GAS, including normal token transfers, deployment of smart contracts, dApp execution, etc.

The Neo N3 Genesis block minted the exact quantity of GAS tokens required to account for all GAS tokens that were in circulation on the Neo Legacy chain at the time of N3's genesis. Slightly over 52 million GAS tokens were generated this way, allowing for all Legacy users to migrate to the new network.

NEP-17 Assets

Like the NEP-5 standard of Neo Legacy, the NEP-17 proposal outlines a token standard for the Neo blockchain that will provide systems with a generalized interaction mechanism for tokenized smart contracts. NEP-17 tokens are issued and managed through smart contracts using an account model, with balance and other token information stored in the smart contract storage.

One notable improvement over the original NEP-5 standard is the onPayment functionality, which allows smart contracts to react to token payments with arbitrary logic. This prevents the need for allowance functions, ensuring users always remain in control of their own assets. It also makes life simpler for developers, granting greater flexibility to smart contracts on N3 in terms of asset management, including the ability to reject unwanted transfers outright, preventing accidental token loss for users.

Smart Contracts

From the blockchain perspective, a smart contract is a set of promises, specified in digital form, including protocols within which the parties perform on these promises.

Cryptography

Encoding Algorithm - Base 58

Base58 is a group of encoding/decoding schemes used to switch data between binary format (hexadecimal) and alphanumeric text format (ASCII). Base58 enables data compressing, is easy to identify, and is suitable for constructing encoding mechanisms of a transmission system that is anti-auto-monitoring. However, lack of verification makes it not able to detect errors during transmission. Thus Base58Check, an improved scheme, is required.

The Base58's alphabet includes numbers (From 1 to 9), and English letters except O (uppercase o) / I (uppercase i) / l (lowercase L). These letters are omitted to avoid confusion.

Neo's Base58 alphabet:
123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz

Base58Check

Base58Check is an improved encoding / decoding method based on Base58. Base58Check solves the lack of checking mechanism in Base58 using the first four bytes of a double SHA256 hash of the data being encoded as a checksum.

Hash Algorithm

A hash function, or hash algorithm, is a method of creating a digital fingerprint from any kind of data. A hash function compresses messages or data into a digested version to shrink the data into a fixed data size. This function disorganizes and remixes data, rebuilding a data fingerprint as a hash value. A hash value is always represented by a short string consisting of random letters and digits.

Two different hash functions are used widely in the Neo system: SHA256 and RIPEMD160. The former is used to generate a longer hash value (32 bytes) and the latter is used to generate a shorter hash value (20 bytes). Usually when a hash value of an object is generated, hash functions are used twice. For example, when a hash of a block or transaction is generated, SHA256 is calculated twice; when a contract address is generated, the SHA256 hash of the script is calculated, then the RIPEMD160 hash of the previous hash is calculated.

In addition, the block will also use a hash structure called a Merkle Tree. It computes the hash of each transaction and combines one with the next and then hashes again, repeating this process until there is only one root hash (Merkle Root).

RIPEMD160

RIPEMD is a cryptographic hash function published by Hans Dobbertin, Antoon Bosselaers Bart Prenee from COSIC research team, University of Leuven in 1996.

RIPEMD160 is a 160-bit improvement based on RIPEMD. This algorithm produces a 160-bit hash, which can be presented in hexadecimal format. One feature of this algorithm is avalanche effect, i.e. any slight changes can result in a totally different hash value.

Neo generates 160-bit hash of contract script with RIPEMD160.

SHA256

SHA256 is a kind of SHA-2 algorithm. SHA-2 is a cryptographic hash function algorithm standard produced by NSA. It belongs to the SHA family. It is a successor of SHA-1. SHA-2 has 6 different algorithm standards, including SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 and SHA-512/256. For any length of message, SHA256 will generate a 256-bit hash value (can be represented by a hexadecimal string).

Murmur32

Murmur is a type of non-cryptographic hash algorithm that suits general hash indexing. It was proposed by Austin Appleby in 2008. Later several derived versions were released. Compared with other popular hash functions, Murmur's random distribution performance is better for keys with high regularity.

Features:

1. Low collision probability
2. Fast computing rate
3. Good performance for large files

Scrypt

Scrypt is a type of secure-cryptographic algorithm based on the PBKDF2-HMAC-SHA-256 algorithm. It was developed by Colin Percival, a famous FreeBSD hacker, for his backup service Tarsnap. The original design was to reduce CPU load, minimize CPU reliance, and use CPU idle time for calculations. Scrypt not only takes a long time to calculate, but also consumes a lot of memory, making it difficult to calculate multiple digests in parallel. So it is more difficult to use the rainbow table for brute-force-attacks.

Neo mainly uses the Scrypt algorithm to generate encrypted secret keys complying with the NEP-2 wallet standard.

ECC Encryption Algorithm

Like Bitcoin, Neo adopts Elliptic Curve Cryptography (ECC) as a public key generating algorithm. The ECC algorithm is a type of asymmetric encryption algorithm. With the irreversible feature of the K=k*G process (K: public key, G: base point (constant)), it can prevent solving a private key from a public key by brute force. With the same length of a secret key, ECC has a higher security level and saves computing resources compared to other encryption algorithms such as RSA. ECC combined with other algorithms, is widely used in signing fields, i.e. ECDSA digital signature.

ECDSA signing

Elliptic Curve Digital Signature Algorithm (ECDSA) is a simulation of the Digital Signature Algorithm (DSA) by ECC algorithm. Some of its advantages include fast speed, reliable strength, and a short signature.

AES Encryption

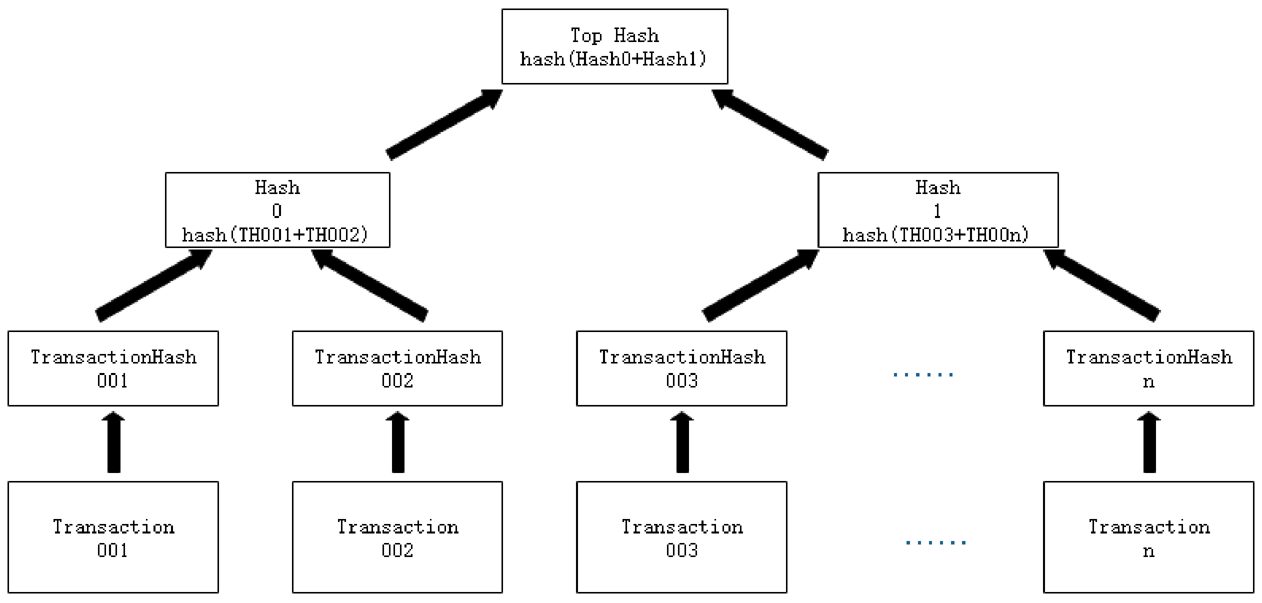
Advanced Encryption Standard (AES) is a type of block cipher algorithm in symmetric encryption algorithms. Its block size can be 128, 192, 256. AES has the following advantages:

1. Fast processing speed
2. Whole process can be described in math
3. Currently no effective cracking method

Neo uses a 256-bit AES encryption algorithm, where encryption mode is ECB and the filling method is NoPadding.

Merkle Tree

Merkle tree is a type of binary tree. It's able to quickly check and induce massive data, and verify the completeness of block transaction records. Neo uses the Merkle tree to construct the block model. Neo's block head stores the Merkle root of all transactions within the block. Block data area stores transaction array.



Attributes of Merkle tree:

1. Merkle tree is basically a binary tree, with all features of a tree structure.
2. Merkle tree's leaf nodes' value is unit data of data set, or unit data hash.

3. The value of a non-leaf node is based on all the leaf node values below it, calculated with a hash method.

Usage Scenarios

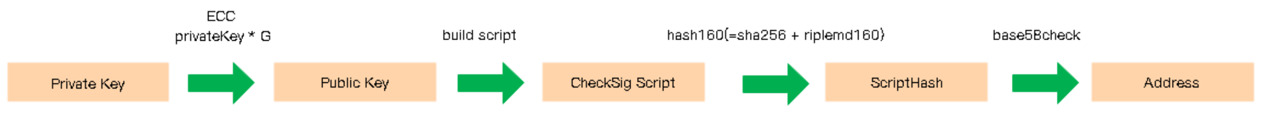
- Builds a Merkle tree root when constructing a block header
- Verifies the block data using SPV wallets.
- As a data structure, it generates a state root for NEO blocks. This is used in cross-chain and light node scenarios for quickly verifying the validity of blocks.

Wallets

Wallets are basic components of Neo and the bridges for users to access the Neo network. They are responsible for transaction operations such as transfering, message signing, contract deployment, etc.

Accounts

In Neo, the account is the smart contract and the address represents a contract script. The below flow diagram shows how to derive the public key from the private key and then to the address:



Private Key

A private key is a random value generated between 1 and n (n is a constant, less than 2^256 slightly), and is generally represented by a 256 bit (32 bytes) number.

There are two main encoding formats for private keys in Neo:

- Hexstring Format

The hexstring format is a string that uses hexadecimal characters to represent a byte array.

- WIF Format

The WIF format is to add prefix 0x80 and suffix 0x01 in the original 32-byte data, and get the string after Base58Check encoding.

Public Key

The public key is a point (X, Y) calculated through the ECC algorithm with the private key. The X, Y coordinates can be represented by 32-byte data. Different from Bitcoin, Neo chooses secp256r1 as the curve of the ECC algorithm. There are two public key formats in Neo:

- Uncompressed Public Key
  0x04 + X (32 bytes) + Y (32 bytes)

- Compressed Public Key
  0x02/0x03 + X (32 bytes)

Address

Address is a string of numbers and letters after a series of transformations of the public key. This section will describe the steps of conversion from a public key to an address in Neo.

Wallet Address Scripthash

When creating a wallet in the Neo blockchain, the private key, public key, wallet address, and related scripthash are generated.

Wallet Files

db3 Files

The db3 wallet is commonly used for exchange wallets to facilitate a large amount of account information storage and the retrieval queries. A db3 wallet file uses SQLite to store data, and the file name extension is .db3. The db3 wallet uses the AES (symmetrical encryption) as its encryption and decryption method.

NEP-6 Files

An NEP-6 wallet file complies with the NEP-6 standard, and the file name extension is .json. An NEP-6 wallet uses the Scrypt algorithm as the core method of wallet encryption and decryption.

Signature

Neo employs the ECDSA algorithm to sign the transaction through the wallet component, taking NIST P256 (secp256r1) as the ECC curve and SHA-256 as the hash algorithm.

Wallet software

Full-node wallet

A full-node wallet is a complete backup of blockchain data, which saves all the on-chain data and participates in the P2P network, therefore it needs a large storage space. Neo-CLI and Neo-GUI are both full-node wallets.

SPV wallet

The SPV (Simplified Payment Verification) wallet is different from a full-node wallet. It doesn't store all block data, only block header data, and verifies the data by using a bloom filter and merkle tree proofs. It's mostly used in mobile apps or light clients, as it effectively reduces storage space requirements.

Transaction

A transaction is the basic operation model of the Neo network. Wallets, smart contracts and accounts interact with the Neo network through transactions. In Neo's P2P network, information is packed for transferring. Different payloads have their own special data.

Structure

On the Neo blockchain the transaction data structure is as follows:

Size	Field	Description
version	byte	Transaction version, currently 0
nonce	uint	Random number
sysfee	long	System fee paid for network resources

netfee	long	Network fee paid for the validator packaging transactions
validUntilBlock	uint	Transaction validity period
signers	Signer[]	Sender and the effective scope of signature
attributes	TransactionAttribute[]	Transaction attributes
script	byte[]	Script executed on the NeoVM
witnesses	Witness[]	List of scripts used to validate the transaction

version

The version allows the transaction structure to be updated to make it backward compatible. The current version is 0.

signers

The first field is the script hash of the transaction sender account. Since the UTXO model has been deprecated in Neo N3 and the native assets NEO and GAS have become NEP-17 assets, the input and outputs fields are no longer recorded in the transaction structure.

The rest of the fields are used to define the effective scope of signature. When checkwitness is used for transaction verification, all cosigners except the transaction sender need to define the scope of their signature.

sysfee

The system fee depends on the transaction's script, i.e., its size, number, and type of NeoVM instructions. The 10 GAS system fee waiver does not exist in Neo N3. The calculation formula is as follows:

$$SystemFee = \sum_{i \in OpcodeSet} OpcodePrice_i * n_i$$

netfee

The network fee is charged when the user submits a transaction to the Neo blockchain as a reward for consensus nodes generating blocks. There is a base fee for each transaction. The transaction is only executed if the fee paid by the user is greater than or equal to the base fee; otherwise, the transaction will be treated as invalid. The calculation formula is as follows:

$$NetworkFee = VerificationCost + tx.Length * FeePerByte$$

attributes

Additional attributes are allowed to be added to transactions of specific types. You need to define the usage type, internal and external data size for each attribute. Up to 16 attributes can be added to one transaction.

script

The script that is executed on the NeoVM and determines the effects of the transaction.

witnesses

Witnesses verify the validity and integrity of a transaction. It includes two attributes. You can add multiple witnesses to each transaction, or use witnesses with multiple signatures.

Invocation Script

An invocation script can be constructed to add a signature. By repeating this step, the invocation script can push multiple signatures for the multi-signature contract.

Verification Script

Verification script, commonly known as address script, includes a normal address script and multi-signature address script. The address script can be directly obtained from the wallet account. It can also be used as a custom authentication contract script.

Transaction Serialization

In Neo all variable-length integer types except IP addresses and port numbers are stored in little-endian order.

Transaction Signature

The transaction signature is to sign the data of the transaction itself by ECDSA method (not including the signature data, i.e. the witnesses) and then fill in the witnesses in the transaction body.

Signature Scope

In Neo Legacy, the transaction signature is globally effective. In order to allow users to control the signature scope at a finer level of granularity, WitnessScope is added to Neo N3 and the signers field in the transaction structure is changed, so that the signature can be used only for verifying a specified contract, preventing unauthorized contracts from using the user signature.

The supported witness scopes are as follows:

- Global
- None
- CalledByEntry
- CustomContracts
- CustomGroups
- Rules

Consensus

Delegated Byzantine Fault Tolerance 2.0

The Byzantine Fault Tolerance (BFT) mechanism is a universal solution for distributed systems. Neo proposes a *delegated* Byzantine Fault Tolerance (dBFT) consensus algorithm based on the Practical Byzantine Fault Tolerance (PBFT) algorithm. The dBFT algorithm determines the validator set according to real-time blockchain voting, which effectively enhances the effectiveness of the algorithm, bringing block time and transaction confirmation time savings. dBFT 2.0 as an upgraded version was released in March 2019, which improves robustness and safety by introducing 3-stage consensus as well as a recovery mechanism.

Single Block Finality of dBFT 2.0

Neo's original dBFT 1.0 algorithm was susceptible to a single block fork in rare cases of network latency. dBFT 2.0 fixes this problem, hence there is no longer any possibility of chain splits.

Consensus Nodes Election

Neo is an open and transparent blockchain network where anyone can either initiate a transaction to apply for being a validator candidate or vote to decide which validator candidate can become a consensus node. The committee members and validators are elected based on the voting result.

Committee members have the privilege to modify the configuration of the Neo network by voting, currently including fee per byte for network transmission, execution fee factor, storage price, blocking/unblocking account, etc.

There is no duty assigned to candidates. However, committee members and validators are elected from a certain number of candidates with the most votes.

Every address has the right to vote to only one candidate address (whether or not it's a committee member). Candidate's received votes are defined as the sum of NEO held by its voter.

Voting is a dynamic and continuous process. If the NEO asset balance of a voter is changed, the number of votes at the previous voting address will also change. The list of consensus nodes and committee members will change accordingly every 21 blocks, known as an epoch.

Genesis Block is the first block, its NextConsensus is set to the script hash of the standby consensus nodes' multi-signature contract.

Governance and Incentives

As a community-driven open platform, Neo N3's on-chain governance model introduces new, essential roles along with an incentive model to ensure that all participants are rewarded accordingly.

Governance Strategy

The new governance model consists of candidates, committee members, and consensus nodes. Among them, the committee is responsible for parameter adjustment of the main net and maintenance of the on-chain environment; Consensus nodes are responsible for packaging transactions and generating blocks. Committee members and consensus nodes are elected from a certain number of candidates with the most votes. Their relationship can be described in the following picture. There is no explicit relationship between committee members and consensus nodes but, as default committee member amount (21) is more than that of consensus nodes (7), generally speaking consensus nodes are a subset of committee members.

Candidates

Any and all individuals or organizations can register to become a candidate and seek votes from voters to become a committee member. After the registration transaction has been recorded on-chain, NEO holders can vote for the candidates they believe are best equipped to make the right decisions for Neo. The votes received by a candidate is calculated every 21 blocks as the sum of NEO tokens on all wallet addresses that have voted for that candidate over the past epoch.

To ensure that committee members are truly supported by the community, elections are only effective when more than 20% of NEO tokens are used to vote. Afterwards, a certain number of candidates with the most votes automatically become committee members and consensus nodes.

Committee

The committee members are elected from candidates with the top 21 most votes. The votes received by a candidate is calculated every 21 blocks, so voters can adjust their voting strategies flexibly according to the performance of the committee.

For any proposal to be approved, over 50% of committee members must reach an agreement before signing a transaction to update blockchain params on-chain.

Moreover, the committee can appoint a number of node roles, including:

- Oracle nodes
- StateRoot consensus nodes
- NeoFS Inner Ring nodes

Consensus Nodes

Amongst the 21 committee members, the top voted seven committee members also serve as consensus nodes to promote transaction activity and optimize the Neo blockchain's security. They have the authority to initiate new block proposals and generate blocks.

Similar to committee members, consensus nodes are refreshed every 21 blocks.

Incentives

Inheriting from Neo Legacy, Neo N3 employs the dual-token mechanism, where NEO is used for governance and GAS is used for utility.

NEO

NEO has a max supply of 100 million tokens and the smallest unit of 1, or in other words, it is not divisible. NEO holders are the owners and managers of the Neo network. By executing voting transactions on the Neo network, they can exercise management power by electing committee nodes, and can also claim additional GAS based on the amount of NEO they hold.

GAS

GAS is the fuel token for the realization of Neo network resource control, with a smallest unit of 0.00000001. Users can obtain GAS either through a claim or purchase. When using the Neo network, they need to pay a certain amount of GAS as network fees, such as transfer, registering assets, publishing assets, running DApps, etc.

Unlike the Neo Legacy, there is no supply limit for Neo N3 GAS, and the system fee for transactions will be burned, reducing the circulating supply.

GAS Distribution Rule

In the initial configuration, 5 GAS tokens will be generated per block—this in turn will be distributed to the Neo Committee (consisting of consensus nodes and committee nodes), NEO voters, and all NEO holders. The generated GAS will be distributed according to certain rules shown below:

NEO Holders – 10%

As with Neo Legacy, this portion of GAS is distributed to NEO holders. It is calculated and distributed to NEO holders' wallets according to the NEO holding period, only after the NEO holder has completed a NEO transfer or performed a vote.

Voters – 80%

The vast majority of GAS generated will be used to incentivize NEO holders to vote for committee members. Only those who successfully vote for the elected committee members can receive this part of the reward, which is calculated and distributed during each epoch (21 blocks). In other words, this portion is divided by 28 (21 for committee members, and 7 for consensus nodes). NEO holders who vote for any elected consensus node are rewarded with 2/28 of this portion; NEO holders who voted for any elected committee member which is not a consensus node receives 1/28 of this portion.

Committee and Consensus Nodes – 10%

The remaining 10% of GAS is used to reward 21 committee members for their contributions towards managing and governing the Neo blockchain. Every 21 blocks (known as an Epoch) votes for committee members are recalculated and the incentive shares are redistributed in turn to new members. In addition, the speaker receives network fees for the transactions contained in the current block.

NeoVM

Introduction

NeoVM is a lightweight virtual machine for executing Neo smart contracts. As the core component of Neo, NeoVM has Turing completeness and high consistency, which can implement arbitrary execution logic and ensure consistent execution results of any node in a distributed network, providing strong support for decentralized applications.

With the help of NeoCompiler, source code written in high-level languages including C#, Java, Python, and Go can be compiled into a unified NeoVM instruction set, thus achieving cross-platform. This lowers the development threshold for enabling developers of all backgrounds to participate in the decentralized application development in the Neo ecosystem without learning a new development language.

In addition, NeoVM is highly decoupled from the upper-level code and is customizable by using techniques such as interop services. NeoVM can be used by simply creating an instance, allowing for its application to various blockchain and non-blockchain scenarios.

Infrastructure and Execution Process

Infrastructure

The NeoVM architecture is mainly composed of the execution engine, stack, and interoperation service layer.

ExecutionEngine

ExecutionEngine is the core of NeoVM, mainly responsible for loading scripts and executing corresponding instructions, such as flow control, stack operation, bit operation, arithmetic operation, logical operation, cryptography, etc. It can also interact with external data via an interoperable service layer accessed through a system call.

Stack

NeoVM is a stack-based virtual machine. NeoVM has three types of stack: InvocationStack, EvaluationStack, and ResultStack.

- InvocationStack is used to store all execution contexts of current NeoVM, which are isolated from each other in the stack. Context switching is performed based on the current context and entry context. The current context points to the top element of the invocation stack, which is ExecutionContext0 in the architecture figure. The entry context points to the tail element of the invocation stack, which is ExecutionContextN in the architecture figure.
- EvaluationStack is for storing the data used by the instruction in the execution process. Each execution context has its own evaluation stack.
- ResultStack is used to store the execution result after all scripts are executed.

Interoperation Surface Layer

The interoperation service layer is a bridge between VM and external data. By invoking interoperation interfaces, NeoVM can access the block information, transaction information, contract information, asset information, and other data required for the execution of smart contracts.

Each Neo smart contract can choose whether to make use of a private storage area, which stores data in key-value format. With the help of an interoperation service layer, NeoVM can dynamically modify the corresponding data in a storage area when executing the smart contract.

Also included in the interoperation service layer are encryption algorithms, zero-knowledge proof, network resource access, and more. The growing featureset is designed to meet the needs of developers and streamline the construction of advanced applications.

In addition, the interoperation service layer also supports custom extensions and modifications to meet the customization needs of developers.

Execution Process

Neo supports multi-language smart contract development. NeoCompiler compiles multi-language smart contracts into unified nvm bytecode files, which are then decoded and executed by NeoVM. Cross-platform compatibility is achieved with multi-language compilers and virtual machines.

Introduction of New Technologies on Neo N3

Multi-Language Support

As with any new paradigm, blockchain development comes with a learning curve. Rather than introducing additional overhead, Neo helps developers by integrating seamlessly with the world's most widely used languages and tools such as C#, Python, TypeScript, Java, and Golang.

Neo Native Oracle

With Neo's native oracle service, it becomes trivial for developers to tap into any external resources. Smart contracts are a revolutionary tool, but their usefulness is naturally limited by the data they can access. The native oracle service introduced in N3 allows developers to access rich off-chain data sources through HTTPS or NeoFS requests, opening up new opportunities to create valuable decentralized services.

Oracle solves the problem that blockchain cannot obtain information from the external network. As a gateway for smart contracts to communicate with the outside world, Oracle opens a window to the outside world for blockchain. Oracle nodes jointly verify the data fetched from the network, then smart contracts query the result in the response transactions on the chain.

Neo Oracle Service is an out-of-chain data access service built into Neo N3. It allows users to request the external data sources in smart contracts, and Oracle nodes designated by the committee will access the specified data source then pass the result in the callback function to continue executing the smart contract logic.

Oracle Fees

Neo Oracle Service charges users by the number of requests, 0.5 GAS by default for each. Users must pay additional fees for the response callback function. All fees will be paid when the Request is created. Request fees can be adjusted by the Neo Council.

NeoFS Decentralized Storage

Developer data is securely encrypted and available. Developers will maintain full control over where data is placed and how it is accessed. NeoFS phases out centralized dependencies with the most reliable and performant decentralized storage solution. With HTTP and S3 gateways, no rearchitecting is required to integrate existing applications or users with NeoFS.

NeoID Self-Sovereign Identity

Through NeoID, developers can use verifiable credentials to add a layer of authentication to applications as well as customize the precise level of attribute verification needed to ensure that a contract operates correctly. This includes regulatory compliance and bot filters. Through its open claim issuance and validation tools, users can remain in full control of their personal information.

Domain Name Service

With Neo's domain name service, users can interact easily with the next generation of decentralized services. Neo's native name service provides domain-mapping for a number of record types. Powered by NFTs to enable open trading, NNS aliases can be used to replace public addresses, contract hashes, and NeoFS containers with human-friendly names.

Advanced Interoperability

As a founding member of the interoperability protocol alliance Poly Network, Neo makes cross-chain interaction as easy as a native transaction. Connected with an ever-growing list of other blockchains, applications on Neo are not limited to interacting with native assets or contracts, but can easily tap into resources from heterogeneous networks.

Conclusion

Neo is a joint effort by community groups from all over the world who have come together to establish and develop the building blocks for the next generation internet. After four years of stable MainNet operation, Neo is undergoing its biggest evolution as it migrates to N3–the most powerful and feature-rich version of the Neo blockchain to date.

With one block finality, the dBFT consensus mechanism guarantees fast and efficient finality in a single block. With Oracle, developers are enabled secured access to any off-chain data. With NeoFS, developers gain access to a distributed data storage solution made for scalability and privacy. With multi-language support, developers can write smart contracts in their preferred language such as C#, Go, Python, Java, or TypeScript. With Neo Name Service, developers have access to a decentralized .neo web domain for next-gen internet applications. With Poly.Network-enabled cross-chain interoperability, developers can utilize and work with other blockchains such as Ethereum, Binance Chain, and more. With NeoID, developers now have access to a set of self-sovereign decentralized identity solution standards. With a unique dual token model, governance is separated from utility. With NeoVM, developers can ensure consistent execution results of any node in a distributed network, providing strong support for decentralized applications.

Neo is new again.

All in one, all in Neo.